Recommender System through Inverse Optimal Transport

Lucas Marandat, Théophane Vallaeys, Guillaume Kunsch (MatrixT)

October 2022

1 Context and problem statement

Optimal transport is a mathematical subject involving a wide range of areas, including differential geometry, partial differential equations, optimization, and probability theory. It has been studied since Monge but modern framework of the theory can be be used in larger settings than what was possible back in Monge's. Recommender system is one of those settings.

1.1 About Optimal Transport

In this section, we present a brief formulation of optimal transportation in the discrete settings (but we could define it in a continuous setting as well).

Given two probability vectors $\mu \in \mathbb{R}^m$ and $\nu \in \mathbb{R}^m$, we define:

$$\Pi(\mu, \nu) = \{\pi \in \mathbb{R}_+^{m \times n} | \pi 1 = \mu, \pi^T 1 = \nu\} \tag{1}$$

In other words, π is the joint probability law whose marginals are μ and ν. $U(\mu, \nu)$ is a convex, bounded and closed set. Furthermore if we assume we know a cost matrix $C = [C_{ij}] \in \mathbb{R}^{m \times n}$ where $[C_{ij}]$ measure the cost of moving a unit mass from μ_i to ν_j, then we can formulate the optimal transport problem as the following optimization problem:

$$d(C, \mu, \nu) = \min_{\pi \in \Pi(\mu,\nu)} \langle C, \pi \rangle \tag{2}$$

where $\langle \cdot, \cdot \rangle$ is the Frobenius inner product for matrices.

This quantity describes how to optimally redistribute μ to ν so that the total cost is minimized, hence providing a means to measure the similarity between the two distributions. The minimizer π is called the optimal transport plan.

As in many optimization problem, it is possible to add regularization by introducing discrete entropy in the previous equation, adding a term $-\varepsilon H(\pi)$. This makes the problem convez and the solution unique. In that case, the theoretical and computational treatment is made easier and in particular using the Sinkhorn-Knopp algorithm (Sinkhorn and Knopp (1967)) it is possible to compute a semi-closed solution of π.

1.2 How to use it for recommender system ?

In the case of recommender system, the setting is bit different. Indeed, we don't have access to the cost matrix C but to an observed, noisy and incomplete, matching matrix $\hat{\pi}$. In that case the idea is to reconstruct a cost matrix C from $\hat{\pi}$ and to then use the optimal transport introduced in 1.1 to get π. As a result the overall problem with a regularization function $R(c)$ is

$$\min_{\pi^C} KL(\hat{\pi} || \pi^C) + \varepsilon^{-1} R(c) \tag{3}$$

$$\text{with } \pi^C = \arg \min_{\pi \in \Pi(\mu,\nu)} \langle C, \pi \rangle - \varepsilon H(\pi) \tag{4}$$

and it can be solved through various treatments. The one we propose comes from Ma et al. (2020). What the paper proves is that this constrained problem is equivalent to the following unconstrained problem :

$$\min_{\alpha,\beta,C} \{R(C) - \langle \alpha, \mu \rangle - \langle \beta, \nu \rangle + \langle C, \hat{\pi} \rangle + s(\alpha, \beta, C)\} \tag{5}$$

$$\text{with } s(\alpha, \beta, C) = \varepsilon \sum_{i,j} \exp((\alpha_i + \beta_j - c_{ij})/\varepsilon) \tag{6}$$

The idea of 5 is to equal the problem (3) to another one which takes 3 variables, one of them being C. This new problem can then be solved using repeated convergence on those 3 variables, as displayed below, which is only an alteration of the Sinkhorn-Knopp algorithm. For more information, we refer the reader to the original paper.

Algorithm 1 Matrix Scaling Algorithm for Cost Learning in Discrete Inverse OT (18)

```
Input: Observed matching matrix $\hat{\pi} \in \mathbb{R}^{m\times n}$ and its marginals $\mu \in \mathbb{R}^m, \nu \in \mathbb{R}^n$.
Initialize: $\alpha \in \mathbb{R}^{m\times 1}, \beta \in \mathbb{R}^{n\times 1}, u = \exp(\alpha/\varepsilon), v = \exp(\beta/\varepsilon)$, $c \in \mathbb{R}^{m\times n}$.
repeat
    $K \leftarrow e^{-c/\varepsilon}$
    $u \leftarrow \mu/(Kv)$
    $v \leftarrow \nu/(K^\top u)$
    $K \leftarrow \hat{\pi}/(uv^\top)$
    $c \leftarrow \text{prox}_{\gamma R}(-\varepsilon \log(K))$
until convergent
Output: $\alpha = \varepsilon \log u$, $\beta = \varepsilon \log v$, $c$
```

2 Our approach

We will try to recommend movies to users using the dataset Movielens (ml-latest-small) composed of 9,724 movies ($m = 9724$) and 610 users ($n = 610$) with 100,836 ratings.

2.1 Extracting features

2.1.1 Using matrix factorization

The first step in the method is to extract features from users and matrix. There are various ways of doing that. For instance, we could use tags already in the dataset, create new ones based on expert knowledge, use a neural network for embeddings. Here we propose to do it through matrix factorization. Let's note $M \in \mathbb{R}^{m\times n}$ the recommendation matrix observed. The problem is finding low-rank matrix $V \in \mathbb{R}^{m\times k}$ and $U \in \mathbb{R}^{n\times k}$, with $k \ll m, n$ such that:

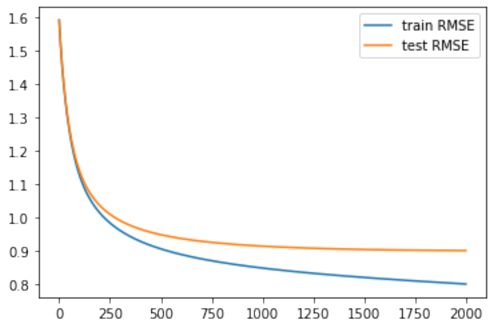
$$\min_{U,V} \|M - VU^T\| \tag{7}$$

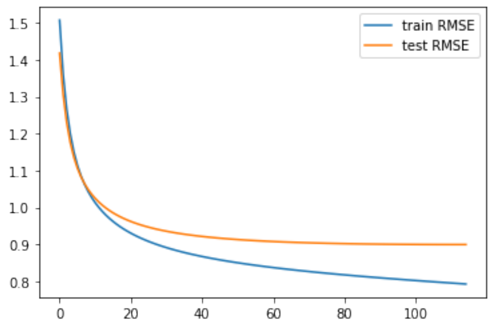
One constraint of this method is that it imposes to have the same number of features for users and movies. The linear property means it is limited, but easy to use. As we have a very sparse matrix in input, we used a further optimization to compute U and V, that enhances greatly the performances, and solves the problem of the *zeros* of the matrix.

If we denote by $X = \{(u_i, v_i)\}$ the points such that $M_{u_i,v_i} \neq 0$, then we will solve the following problem using gradient descent (with the Adam optimizer):

$$\min_{U,V} \sum_{i=0}^{|X|} (U_{u_i}^\top V_{v_i} - M_{u_i,v_i})^2 \tag{8}$$

We implemented matrix factorization using PyTorch. We tested various number of features between 2 and 75, and it appears 10 was the best one. We also implemented it with batches to optimize computational cost.





With Gradient Descent on the whole batch (on the left), with k = 10 and 2,000 steps we obtain a test RMSE (Root Mean Square Error) of 0.9, on scale from 0 to 5 for human ratings. With Gradient Descent with batches of size (on the right), with k = 10 and 115 steps we obtain a test RMSE of 0.9, on scale from 0 to 5 for human ratings.

2.1.2 Using a neural network

The idea of using a neural network is to extract more meaningful features: movies will be rated by a lot of users, but each user will only rate a few movies. But each user can also be seen as a combination of the features of every movies they saw, combined with the ratings. We will learn a matrix V of user features and the functions $f_1 : \mathbb{R}^1 \to \mathbb{R}^k$, $f_2 : \mathbb{R}^k \to \mathbb{R}^k$ and $f_3 : \mathbb{R} \to \mathbb{R}$, in order to solve:

$$\min_{U,V} \sum_{i=0}^{|X|} \left(V_{v_i}^\top f_3 \left(\sum_{l \in S_{u_i}, l \neq v_i} f_2(V_l \times f_1(M_{u_i,l})) \right) - M_{u_i,v_i} \right)^2 \tag{9}$$

where S_{u_i} is the set of movies rated by the user u_i. Each function f_i is represented by a small neural network with no hidden layers and a width of k (the number of features). The choice of such a small architecture is the result of ablations on a larger network, with each ablation preserving the results. As such, we pretend to have a small function, easy to train using gradient descent (with Adam optimizer), giving meaningful features. We only regret that the error given by using these features directly, as if computed by MF, is worse than the MF features itself.

2.2 Inverse Optimal Transport

Then we parametrize the cost matrix C by $A \in \mathbb{R}^{k \times k}$ through the formula:

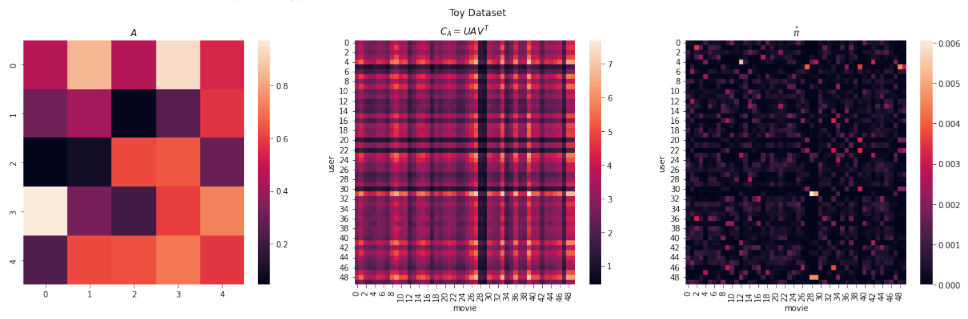
$$C_A = UAV^T \tag{10}$$

We can then resolve the problem of equation (3) using the method presented in 1.2 and $\hat{\pi} = \frac{M}{\sum_{ij} M_{ij}}$. To make a prediction for ratings for test set, we use the optimal C_A matrix find in the training and we resolve problem (2) (or it's regularized version). We then select the movie i and user j (that of course wasn't known in the training set), and use the value of π_{ij} on which we apply a rescaling function to find a real between 0 and 5, similar to a human rating.

We implemented the modelisation presented in Ma et al. (2020) from their Github repository and made the adapted change to make it run locally. For Inverse Optimal Transport, and thus the recommending part in itself, we first tested our results on a toy dataset to understand behavior of the algorithms and how it performed in case where we know the ground truth.

2.2.1 Toy Dataset

For that we generated an interaction matrix A with random numbers, plus random U and V matrix. We want to make sure, if knowing the exact cost matrix, the model is capable of finding it itself. Using this cost matrix, $C_A = UAV^T$, we then generate $\hat{\pi}$ using the Sinkhorn-Knopp algorithm (Sinkhorn and Knopp (1967)).



Using this OT matrix $\hat{\pi}$ with valid features U and V, we then divide it into a training data set and a test data set. After running the above algorithm with the training dataset, we find that our model smooths out the ground truth, and thus does not hold precisely to it. To better understand these results, we compare the accuracy of the algorithm to two baseline algorithms:

- A random algorithm: $Rand := Uniform\,(0, max(\hat{\pi}_{train}))$
- A constant algorithm: $Mean := mean(\hat{\pi}_{train})$
- A Matrix Factorization algorithm: MF

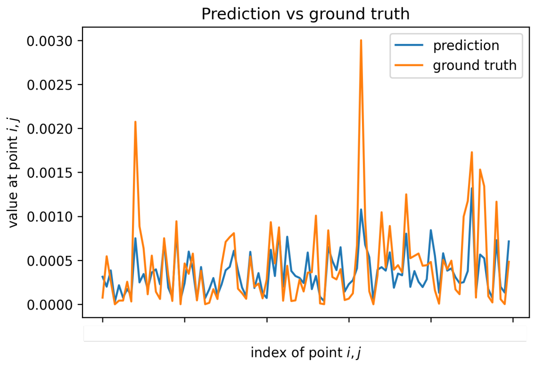


Table 1: Accuracy on the test dataset

$Rand$	$Mean$	MF	IOT
0.0028	0.0006	0.0006	0.0004

These results allow us to conclude that the algorithm indeed converges to a good solution in a perfect context.

2.2.2 MovieLens Dataset

When it comes to the ml-latest-large dataset, for which we do not know the true interaction matrix A, we should expect worse results than before. We first learn two models for extracting user and movie features (U and V respectively):

- MF solving the equation (8)
- $DeepMF$ solving the equation (9)

Using these models, we were able to compare their impact on the final algorithm. After training and tuning the algorithm, the results were not as good as expected on the toy dataset and in the experiences of Ma et al. (2020). Using the same plotting methods as with the toy dataset, we can see that our algorithm performs worse than MF and $DeepMF$, and even worse than $Mean$.

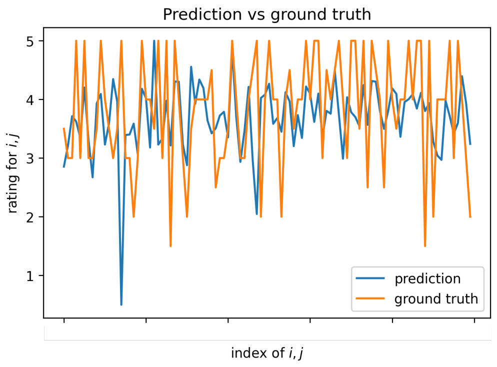


Table 2: Accuracy on the test dataset

$Mean$	MF	$DeepMF$	IOT
0.9347	0.7152	0.7995	0.9661

3 Conclusion

We implemented various techniques for recommender using inverse optimal transport and matrix factorization. Even if matrix factorization brings fairly good results (RMSE of 1 star) for a simple modelisation, the results for the Inverse Optimal Transport, even if theoretically promisings, were not as good as expected on the MovieLens dataset. However, for the toy dataset it was a able to get a good grasp of ground truth.

References

Ma, S., Sun, H., Ye, X., Zha, H., and Zhou, H. (2020). Learning cost functions for optimal transport.

Sinkhorn and Knopp, . (1967). Concerning nonnegative matrices and doubly stochastic matrices. *Pacific J. Math*, 21.